**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №3**

**по дисциплине «Программирование»**

Тема: "**ДВУМЕРНЫЕ СТАТИЧЕСКИЕ МАССИВЫ. УКАЗАТЕЛИ"**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 4373 |  | Андреева П.Д. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2024

**Цель работы.**

Изучение работы с двумерными массивами и сортировками, перестановками и указателями.

**Основные теоретические положения.**

Текстовые строки представляются с помощью одномерных массивов символов. В языке C++ текстовая строка представляет собой набор символов, обязательно заканчивающийся нулевым символом (‘\0’). Поэтому, если вы хотите создать текстовый массив для хранения 10 (N) символов, нужно выделить память под 11(N+1) символов.

Объявленный таким образом массив может использоваться для хранения текстовых строк, содержащих не более 10 символов. Нулевой символ позволяет определить границу между содержащимся в строке текстом и неиспользованной частью строки.

При определении строковых переменных их можно инициализировать конкретными значениями с помощью строковых литералов:

char S1[15] = “This is text”;  
  
char S2[] = “Пример текста”;

Последние два элемента переменной просто не используются, а строка автоматически подстраивается под длину инициализирующего текста.

При работе со строками можно обращаться к отдельным символам строки как в обычном одномерном массиве с помощью индексов:

cout << S1[0]; // На экране будет выведен символ ‘T’

Если строка формируется при помощи цикла (или иного способа), то необходимо в ее конец обязательно записать нулевой символ '\0'.

При выводе строк можно использовать форматирование (манипуляторы или функции потока вывода). Вывод текстовых строк на экран крайне простая задача.

Предпочтительно использование функции потока ввода cin.getline:

const int N = 21;  
  
char Str [N];  
  
cin.getline (Str, N); // Пусть введена строка “Это пример текста”  
  
cout << Str << endl; // На экран будет выведено “ Это пример текста”

Если длина введенного с клавиатуры текста превышает максимальную длину массива , в него будет записано (в нашем примере) 20 символов вводимого текста и нулевой символ. Остальные символы введенного текста остаются во входном потоке и могут быть взяты из него следующими инструкциями ввода.

Функция cin.getline может иметь третий параметр, задающий символ, при встрече которого чтение строки из потока прекращается:

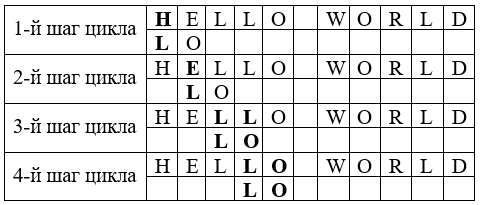
cin.getline (Str, N, ‘.’);

Иногда чтение из потока невозможно (например, попытка считать слишком длинный текст). Для того чтобы продолжить чтение из потока, необходимо восстановить его нормальное состояние. Этого можно достигнуть с помощью функции потока cin.clear(), которая сбрасывает состояние потока в нормальное. Если забирать остатки данных из потока ввода не надо, то следует очистить его с помощью функции cin.sync().

### **Прямой поиск**

При работе со строками часто будет возникать потребность в поиске набора символа или слов (поиска подстроки в строке). При условии, что текст может быть крайне большим, хочется, чтобы алгоритм поиска подстроки работал быстро.

Самый простой способ подстроки в строке – Линейный поиск – циклическое сравнение всех символов строки с подстрокой. Действительно, этот способ первый приходит в голову, но очевидно, что он будет самым долгим.



На первых двух итерациях цикла сравниваемые буквы не будут совпадать. На третьей же итерации, совпал символ ‘L’, это означает, что теперь нужно сравнивать следующий символ подстроки со следующим символом строки. Видно, что символы отличаются, поэтому алгоритм продолжает свою работу. На четвертой же итерации подстрока была найдена.

Если представить, что исходная строка непорядок больше и подстрока находится в конце строки (или вовсе отсутствует), то сразу видны минусы данного алгоритма.

Одной из вариаций алгоритма полного перебора является реализация таблицы включений. Суть алгоритма: дана подстрока S и строка T. Требуется определить индекс, начиная с которого образец S содержится в строке T. Если S не содержится в T, необходимо вернуть индекс, который не может быть интерпретирован как позиция в строке.

### **Бойера-Мура**

Алгоритм Бойера-Мура считается наиболее быстрым среди алгоритмов общего назначения, предназначенных для поиска подстроки в строке.

Преимущество этого алгоритма в том, что ценной некоторого количества предварительных вычислений над подстрокой (но не над исходной строкой, в которой ведётся поиск), подстрока сравнивается с исходным текстом не во всех позициях (пропускаются позиции, которые точно не дадут положительный результат).

Поиск подстроки ускоряется благодаря созданию таблиц сдвигов. Сравнение подстроки со строки начинается с последнего символа подстроки, а затем происходит прыжок, длина которого определяется по таблице сдвигов. Таблица сдвигов строится по подстроке так чтобы перепрыгнуть максимальное количество символов строки и не пропустить вхождение подстроки в строку.

Правила построения таблицы сдвигов:

1) Значение элемента таблицы равно удаленности соответствующего символа от конца шаблона (подстроки).

2) Если символ встречается более одного раза, то применятся значение, соответствующее символу, наиболее близкому к концу шаблона.

3) Если символ в конце шаблона встречается 1 раз, ему соответствует значение, равное длине образа; если более одного раза – значение, соответствующее символу, наиболее близкому к концу образа.

4) Для символов, отсутствующих в образе, применяется значение, равное длине шаблона.

**Постановка задачи.**

Необходимо написать программу, которая реализует поставленную задачу:

1) С клавиатуры или с файла (\*) (пользователь сам может выбрать способ ввода) вводится последовательность, содержащая от 1 до 50 слов, в каждом из которых от 1 до 10 строчных латинских букв и цифр. Между соседними словами произвольное количество пробелов. За последним символом стоит точка.

2) Необходимо отредактировать входной текст:

· удалить лишние пробелы;

· удалить лишние знаки препинания (под «лишними» подразумевается несколько подряд идущих знаков (обратите внимание, что «…» - корректное использование знака) в тексте);

· исправить регистр букв, если это требуется (пример некорректного использования регистра букв: пРиМЕр);

3) Выполнить задание по варианту:

|  |  |
| --- | --- |
| 1 | Вывести на экран слова последовательности в обратном порядке. |
| 2 | Вывести на экран слова последовательности в алфавитном порядке. |
| 3 | Вывести на экран слова последовательности, не содержащие цифр. |
| 4 | Вывести на экран только те слова последовательности, в которых встречаются одинаковые буквы. |
| 5 | Вывести на экран только те слова последовательности, в которых первая буква слова встречается в этом слове еще раз. |
| 6 | После окончания ввода последовательности вывести на экран сначала все слова, содержащие только буквы, затем слова, содержащие только цифры, а потом слова, содержащие и буквы, и цифры. |

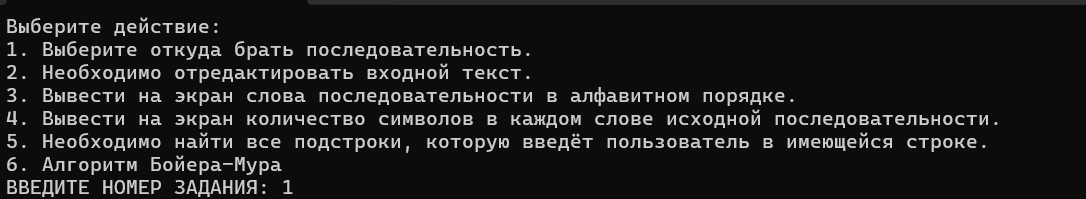
4) Выполнить задание по варианту:

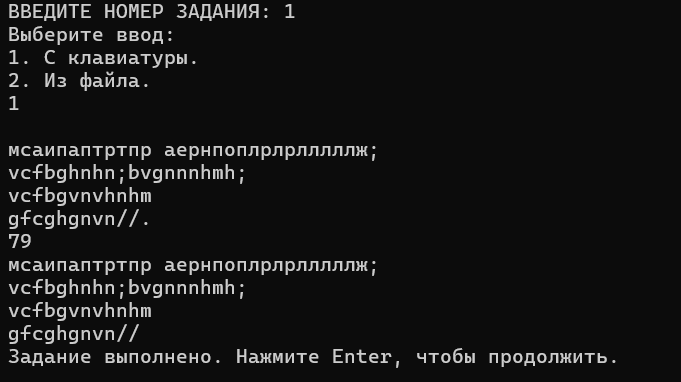
|  |  |
| --- | --- |
| 1 | Вывести на экран ту же последовательность, заменив во всех словах первую букву соответствующей прописной буквой. |
| 2 | Вывести на экран количество символов в каждом слове исходной последовательности. |
| 3 | Вывести на экран ту же последовательность, удалив из всех слов заданный набор букв и (или) цифр. |
| 4 | Вывести на экран ту же последовательность, заменив во всех словах цифры на буквы латинского алфавита, номера которых в алфавите равны заменяемой цифре. |
| 5 | Вывести на экран ту же последовательность, переместив все цифры, содержащиеся в словах, в конец соответствующих слов. |
| 6 | Вывести все слова исходной последовательности на экран вертикально. |
| 7 | Вывести на экран все слова последовательности в две или три колонки (в зависимости от количества слов) с выравниванием слов по правой границе колонки. |

5) Необходимо найти все подстроки, которую введёт пользователь в имеющейся строке. Реализуйте два алгоритма: первый алгоритма – Линейный поиск, а второй алгоритм согласно вашему номеру в списке. Четные номера должны реализовать алгоритм КМП, а нечетные – Бойера-Мура. (\*)

**Выполнение работы.**

Код программы представлен в приложении А





**Выводы.**

Я узнала, как искать подстроки, сортировать их, находить определенные элементы строк, выводить и менять значения в строке, редактировать текст, читать текст из файла.

Приложение А

рабочий код

#include <iostream>

#include <windows.h>

#include <fstream>

using namespace std;

unsigned my\_str\_len(char\* S)

{

unsigned L = 0;

while (S[L])

++L;

return L;

}

void keyboard\_input(char\* text, int N) {

cin.getline(text, N, '.');

int len\_of\_sequence = my\_str\_len(text);

for (int j = 0; j < len\_of\_sequence; j++) {

text[j] = text[j + 1];

}

--len\_of\_sequence;

cin.sync();

}

void input\_from\_file(char\* text) {

ifstream File;

File.open("text.txt");

if (!File.is\_open()) {

cout << "" << endl;

}

File.read((char\*)text, sizeof(text));

File.close();

}

void print\_result(char\* text, int N) {

for (int i = 0; i < N; ++i)

cout << text[i];

cout << endl;

}

void delete\_space(char\* text, int N) {

for (int i = 0; i < N; ++i) {

if (text[i] == ' ' and text[i + 1] == ' ') {

while (text[i + 1] == ' ') {

for (int j = i + 1; j < N; j++) {

text[j] = text[j + 1];

}

--N;

}

}

}

}

void delete\_punctuation(char\* text, int N) {

for (int i = 0; i < N; ++i) {

if (ispunct(text[i]) and ispunct(text[i + 1])) {

if (text[i] == '.' and text[i + 1] == '.' and text[i + 2] == '.' and not ispunct(text[i + 3])) {

i += 3;

}

else {

while (ispunct(text[i + 1])) {

for (int j = i + 1; j < N; j++) {

text[j] = text[j + 1];

}

--N;

}

}

}

}

}

void make\_lower(char\* text, int N) {

for (int i = 0; i < N; ++i) {

if (i == 0) {

text[i] = toupper(text[i]);

}

else if (text[i - 1] == ' ' and isalpha(text[i])) {

text[i] = toupper(text[i]);

}

else {

text[i] = tolower(text[i]);

}

}

}

void correctLetterCase(char\* text) {

for (int i = 0; i < my\_str\_len(text); i++) {

if (isalpha(text[i]) && (i == 0 || text[i - 1] == ' ')) {

text[i] = toupper(text[i]);

}

else {

text[i] = tolower(text[i]);

}

}

}

void how\_many\_symbols\_in\_word(char\* text, int N) {

int counter = 0;

for (int i = 0; i < N; i++) {

if (text[i] != ' ' && text[i] != '\0') {

cout << text[i];

counter += 1;

}

else {

cout << " - " << counter << endl;

counter = 0;

}

}

}

int linear\_search(char\* text, char pattern[], int N) {

int pattern\_len = my\_str\_len(pattern);

for (int i = 0; i <= N - pattern\_len; ++i) {

int j;

for (j = 0; j <= pattern\_len; ++j) {

if (text[i + j] != pattern[j]) {

break;

}

}

if (j == pattern\_len) {

return i;

}

}

return -1;

}

void prepare\_table(const char\* pattern, int pattern\_len, int table[256]) {

for (int i = 0; i < 256; i++) {

table[i] = pattern\_len;

}

for (int i = 0; i < pattern\_len - 1; i++) {

table[(unsigned char)pattern[i]] = pattern\_len - i - 1;

}

}

int Boyer\_Moore\_algorithm(char\* text, char pattern[], int N, int\* table) {

int pattern\_len = my\_str\_len(pattern);

int text\_len = my\_str\_len(text);

prepare\_table(pattern, pattern\_len, table);

int j, k;

int i = pattern\_len - 1;

while (i < text\_len) {

int j = pattern\_len - 1;

int k = i;

while (j >= 0 && k >= 0 && text[k] == pattern[j]) {

k--;

j--;

}

if (j < 0) {

return k + 1;

}

i += table[(unsigned char)text[i]];

}

return -1;

}

int organize\_words(char\* text, char words[][10]) {

int i = 0, word\_len = 0, count = 0;

while (text[i] != '\0' && count < 50) {

if (text[i] == ' ' || text[i + 1] == '\0') {

if (word\_len > 0) {

words[count][word\_len] = '\0';

count++;

word\_len = 0;

}

}

else {

words[count][word\_len] = text[i];

word\_len++;

}

i++;

}

return count;

}

void swap\_words(char words[][10], int j) {

char temp[10];

int k = 0;

while (words[j][k] != '\0' && k < 10) {

temp[k] = words[j][k];

k++;

}

temp[k] = '\0';

k = 0;

while (words[j - 1][k] != '\0' && k < 10) {

words[j][k] = words[j - 1][k];

k++;

}

words[j][k] = '\0';

k = 0;

while (temp[k] != '\0' && k < 10) {

words[j - 1][k] = temp[k];

k++;

}

words[j - 1][k] = '\0';

}

void sort\_alphabetically(char words[][10], int count) {

int c = 0;

for (int i = 0; i < count - 1; i++) {

for (int j = count - 1; j > i; j--) {

if (words[j][0] < words[j - 1][0]) {

swap\_words(words, j);

}

else if (words[j][0] == words[j - 1][0]) {

while (words[j][c] == words[j - 1][c]) {

c++;

}

if (words[j][c] < words[j - 1][c]) {

swap\_words(words, j);

}

}

}

}

}

void display\_menu() {

cout << "Выберите действие:" << endl;

cout << "1. Выберите откуда брать последовательность." << endl;

cout << "2. Необходимо отредактировать входной текст." << endl;

cout << "3. Вывести на экран слова последовательности в алфавитном порядке." << endl;

cout << "4. Вывести на экран количество символов в каждом слове исходной последовательности." << endl;

cout << "5. Необходимо найти все подстроки, которую введёт пользователь в имеющейся строке." << endl;

cout << "6. Алгоритм Бойера-Мура" << endl;

}

int main()

{

setlocale(LC\_ALL, "");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int identificator, num1, num2, n, len\_of\_sequence, result, count, index;

const int N = 600;

char sequence[N] = { ' ' }, pattern[30], pattern1[30], words[50][10];

int table[256];

while (true) {

display\_menu();

cout << "ВВЕДИТЕ НОМЕР ЗАДАНИЯ: ";

cin >> identificator;

switch (identificator) {

case 1:

cout << "Выберите ввод:" << endl;

cout << "1. С клавиатуры." << endl;

cout << "2. Из файла." << endl;

cin >> num1;

cout << endl;

switch (num1) {

case 1:

keyboard\_input(sequence, N);

sequence[my\_str\_len(sequence)] = ' ';

sequence[my\_str\_len(sequence) + 1] = '\0';

break;

case 2:

const char\* filePath = "C:/laba4/ladna.txt";

ifstream file(filePath, ios::in);

if (!file.is\_open()) {

cout << "Файл не открывается!!!!! \n";

return -1;

}

file.read((char\*)sequence, sizeof(sequence));

file.close();

break;

}

cout << my\_str\_len(sequence) << endl;

print\_result(sequence, my\_str\_len(sequence));

break;

case 2:

cout << "Измененный текст, очищенный от пробелов, знаков препинания и неправильного регистра" << endl;

delete\_space(sequence, my\_str\_len(sequence));

delete\_punctuation(sequence, my\_str\_len(sequence));

make\_lower(sequence, my\_str\_len(sequence));

cout << my\_str\_len(sequence) << endl;

print\_result(sequence, my\_str\_len(sequence));

break;

case 3:

cout << "Отсортированные по алфавиту слова:" << endl;

count = organize\_words(sequence, words);

sort\_alphabetically(words, count);

for (int i = 0; i < count; i++) {

cout << words[i] << " ";

}

break;

case 4:

cout << "Вывести на экран количество символов в каждом слове исходной последовательности." << endl;

how\_many\_symbols\_in\_word(sequence, my\_str\_len(sequence));

break;

case 5:

system("cls");

cout << "Введите слово для линейного поиска" << endl;

cin.clear();

cin.sync();

keyboard\_input(pattern, my\_str\_len(pattern));

result = linear\_search(sequence, pattern, my\_str\_len(sequence));

if (result == -1) {

cout << "Нет таких слов в последовательности" << endl;

}

else {

cout << "Есть такая подстрока,начиная с индекса " << result << endl;

}

break;

case 6:

system("cls");

cout << "Введите слово для поиска" << endl;

cin.clear();

cin.sync();

keyboard\_input(pattern1, my\_str\_len(pattern1));

index = Boyer\_Moore\_algorithm(sequence, pattern1, my\_str\_len(sequence), table);

if (index != -1) {

cout << "Первое вхождение найдено на индексе: " << index << endl;

}

else {

cout << "Шаблон не найден." << endl;

}

break;

case 7:

cout << "Выход!" << endl;

return 0;

default:

cout << "Некорректный номер задания." << endl;

break;

}

cout << "Задание выполнено. Нажмите Enter, чтобы продолжить." << endl;

cin.ignore();

cin.get();

}

return 0;

}